A compact encoding for λ-terms in interaction calculus

Anton Salikhmetov

April 16, 2013

Abstract

We give a concise formal definition of an extremely compact encoding for λ-terms straight into the calculus for interaction nets. Reduction on configurations in the resulting interaction system implements β-reduction. We achieve balance between the minimal set of symbols, the minimal net to represent a term, and reasonable efficiency. We also discuss a directed version of the system and its possible implementation using the approach of blind graph rewriting as well as a hypothetical model based on the suggested encoding for the equational theory of the pure untyped extensional λ-calculus.

1 Introduction

There are two extreme ways to encode λ-terms [1] into interaction nets [2]. They pursue conflicting goals. The original optimal one by Lamping [3] implements β-reduction of λ-terms the asymptotically most efficient way. The other one [4] represents λ-terms in an interaction system with the minimal set of symbols, namely the set of interaction combinators [5]. (In fact, there exists a universal interaction system by Bechet [6] with only two agents instead of three, but its interaction rules appear to be too complicated.) However, both encodings have their own downsides. The set of symbols used by Lamping's encoding is practically infinite due to the use of integer labels for agents. The encoding in interaction combinators in turn appears not to be optimal by Levy. Besides, both encodings involve some overhead to represent a λ-term.

First, we present a compact representation of λ-terms pursuing balance between the minimal set of symbols and efficiency of implementation. To define our encoding in a more concise and formal way, we will use the calculus for interaction nets [7] rather than graphical representation. Namely, λ-terms will be mapped into configurations.

Second, we construct a directed version of the resulting interaction system similar to that of directed combinators [5]. Then, we consider a hypothetical effective model for the equational theory $\lambda K\beta\eta$ based on the directed encoding. Finally, we discuss a possible implementation of the directed encoding using the approach of blind graph rewriting [8]. The non-directed version of our encoding for λ-terms is less preferred for implementation in blind graph rewriting due to an extreme cost of pattern matching in blind graph rewriting which by definition lacks an explicit branching operation if–then–else.

2 Encoding λ-terms

We work in interaction calculus [7].

Let $\lambda, \psi, \delta, \epsilon \in \Sigma$, $\mathrm{Ar}(\lambda) = \mathrm{Ar}(\psi) = \mathrm{Ar}(\delta) = 2$, and $\mathrm{Ar}(\epsilon) = 0$.

For any $\alpha \in \Sigma$, $\beta \in \{\psi, \delta\}$, $\alpha \neq \beta$, and $\mathrm{Ar}(\alpha) = n$, we assume

$$\alpha[x_1, \dots, x_n] \bowtie \alpha[x_1, \dots, x_n];$$
$$\alpha[\delta(x_1, y_1), \dots, \delta(x_n, y_n)] \bowtie \beta[\alpha(x_1, \dots, x_n), \alpha(y_1, \dots, y_n)];$$
$$\alpha[\epsilon, \dots, \epsilon] \bowtie \epsilon.$$

Any λ-term M [1] can be mapped into a configuration $\langle x \mid \Gamma(M, x)\rangle$ as follows:

$$\begin{aligned}
\Gamma(y, x) &= \{x = y\};\\
\Gamma(\lambda y.M, x) &= \{x = \lambda(\epsilon, z)\} \cup \Gamma(M, z), \quad \text{when } y \notin \mathrm{FV}(M);\\
\Gamma(\lambda y.M, x) &= \{x = \lambda(y, z)\} \cup \Gamma(M, z), \quad \text{when } y \in \mathrm{FV}(M);\\
\Gamma(M\ N, x) &= \{y = \lambda(z, x), t_1 = \psi(t'_1, t''_1), \dots, t_n = \psi(t'_n, t''_n)\} \cup \Gamma(M', y) \cup \Gamma(N', z),
\end{aligned}$$

where

$$\begin{aligned}
\{t_1, \dots, t_n\} &= \mathrm{FV}(M) \cap \mathrm{FV}(N);\\
M' &= M[t_1 := t'_1] \dots [t_n := t'_n];\\
N' &= N[t_1 := t''_1] \dots [t_n := t''_n].
\end{aligned}$$

One may check that the normal form of the configuration representing a λ-term will correspond to the β-normal form of the term if any. So, we have implemented $\lambda K \beta$ system in interaction calculus and therefore in interaction nets, too.

3 A directed version of the encoding

The above encoding has balance between the minimal set of symbols, the minimal net to represent a term, and efficiency of the reduction implementation. It has completely no overhead to represent applications, abstractions, and sharing. Besides, we can see that the interaction rules have unified form similar to those for directed combinators [5].

Directed combinators have some advantages over the usual interaction combinators from the viewpoint of software implementation. Specifically, an interaction rule is easier to choose in that system, because the active pairs of directed combinators can be represented as ordered pairs, unlike unordered active pairs in usual interaction systems.

In [5], interaction combinators are represented using directed combinators with two directed combinators per each interaction combinator. However, the directed version of our interaction system can be constructed without doubling the agents to encode λ-terms. Instead, we use a different technique to preserve polarity between the ports of agents. Namely, let $\Sigma = \{\lambda, \lambda^*, \delta, \delta^*, \psi, \psi^*\}$, $\forall \alpha \in \Sigma : \mathrm{Ar}(\alpha) = 2$, and the set $\mathcal{R}$ of interaction rules have the following nine elements:

$$\lambda[a, b] \bowtie \lambda^*[a, b], \qquad \delta[a, b] \bowtie \delta^*[a, b], \qquad \psi[a, b] \bowtie \psi^*[a, b];$$
$$\lambda[\delta(a, b), \delta^*(c, d)] \bowtie \delta^*[\lambda(a, c), \lambda(b, d)], \qquad \lambda[\delta(a, b), \delta^*(c, d)] \bowtie \psi^*[\lambda(a, c), \lambda(b, d)];$$
$$\delta[\lambda^*(a, b), \lambda^*(c, d)] \bowtie \lambda^*[\delta^*(a, c), \delta(b, d)], \qquad \psi[\lambda^*(a, b), \lambda^*(c, d)] \bowtie \lambda^*[\delta^*(a, c), \delta(b, d)];$$
$$\psi[\delta^*(a, b), \delta^*(c, d)] \bowtie \delta^*[\psi(a, c), \psi(b, d)], \qquad \delta[\psi^*(a, b), \psi^*(c, d)] \bowtie \psi^*[\delta(a, c), \delta(b, d)].$$

We omit the erasing agent ϵ which only performs garbage collection in disconnected nets.

One can notice duality between the agents λ, δ, ψ and $\lambda^*, \delta^*, \psi^*$, respectively. That is why we produce two dual encodings for an arbitrary λ-term, namely configurations $\langle x \mid \Gamma(M, x)\rangle$ and $\langle x \mid \Gamma^*(M, x)\rangle$, where the Γ and Γ^* mappings are defined as follows:

$$\begin{aligned}
\Gamma(y, x) = \Gamma^*(y, x) &= \{x = y\};\\
\Gamma(\lambda y.M, x) &= \{x = \lambda(y, z)\} \cup \Gamma(M, z);\\
\Gamma^*(\lambda y.M, x) &= \{x = \lambda^*(y, z)\} \cup \Gamma^*(M, z);\\
\Gamma(M\ N, x) &= \{y = \lambda^*(z, x), t_1 = \psi^*(t'_1, t''_1), \dots, t_n = \psi^*(t'_n, t''_n)\} \cup \Gamma(M', y) \cup \Gamma(N', z),\\
\Gamma^*(M\ N, x) &= \{y = \lambda(z, x), t_1 = \psi(t'_1, t''_1), \dots, t_n = \psi(t'_n, t''_n)\} \cup \Gamma^*(M', y) \cup \Gamma^*(N', z),
\end{aligned}$$

where

$$\begin{aligned}
\{t_1, \dots, t_n\} &= \mathrm{FV}(M) \cap \mathrm{FV}(N);\\
M' &= M[t_1 := t'_1] \dots [t_n := t'_n];\\
N' &= N[t_1 := t''_1] \dots [t_n := t''_n].
\end{aligned}$$

These two dual encodings were found through the following observation: when using unordered representation of λ-terms, the orientation of all application and sharing agents with respect to abstraction agents is preserved during evaluation.

4 Hypothesis: an extensional model

There is a notion of so-called observational equivalence between interaction nets; see [9] and [10]. Two nets are considered bisimilar or observationally equal if they produce the same results in any computable context. In a way, observational equivalence corresponds to extensionality in λ-calculus. In extensional λ-calculus, β-reduction is extended to $\beta\eta$-reduction. One may wonder how to implement η-reduction in interaction nets.

In order to check bisimilarity of two nets, we can carry out experiments, connecting their interfaces sequentially to the interfaces of different nets and then evaluating them. If all the possible experiments produce the same results, the nets are called bisimilar. As they cannot be distinguished by any experiment, they can naturally be considered equal.

In our encoding, η-redexes are represented as $\langle x, y \mid x = \lambda(a, b), y = \lambda^*(a, b)\rangle$ which may be checked to be observationally equal to $\langle x, x \mid \varnothing\rangle$. Now, let us consider a net $c = \langle \varnothing \mid \Gamma(M, x) \cup \Gamma^*(N, x)\rangle$ for some combinators M and N, i. e. $\mathrm{FV}(M) = \mathrm{FV}(N) = \varnothing$. If M and N have the same $\beta\eta$-normal form, then, according to the rules of our interaction system, all the agents in c annihilate, thus $c \downarrow \langle \varnothing \mid x_1 = x_1, \dots, x_n = x_n\rangle$ for some $n \geq 1$.

On the other hand, if M and N have different $\beta\eta$-normal forms M' and N', then at some point of evaluation $\langle \varnothing \mid \Gamma(M', x) \cup \Gamma^*(N', x)\rangle$ one of six duplication rules will eventually take place. Many experiments have shown that extensionally different terms either produce infinite reduction sequence starting from c, or produce a net without active pairs, but still with some agents remaining.

This leads us to the following hypothesis: $c \downarrow \langle \varnothing \mid x_1 = x_1, \dots, x_n = x_n\rangle$ for some $n \geq 1$ if and only if $\lambda K\beta\eta \vdash M = N$. If this hypothesis holds true, we have an effective model of the equational theory $\lambda K\beta\eta$ testing two encoded terms with each other.

5 Blind graph rewriting implementation

We are especially interested in implementing the directed version of our interaction system using the approach of blind graph rewriting [8]. For instance, it can be done by using four spaghetti stacks in the heap, each stack containing ordered pairs of nodes. One stack is dedicated to active pairs whose interaction rule is duplication. The second stack is meant for annihilating agents. Two more stacks will represent equations reducible by indirection rule by left and right hand side, respectively.

Having these four stacks, the interaction and indirection rules can be implemented in blind graph rewriting in the obvious way. The only part which is not trivial is how to choose a stack for a new active pair. The latter is supposed to be achieved by if–then–else operation implemented in the blind graph rewriting system as described in [8].

References

[1] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics.* North-Holland, 1984.

[2] Y. Lafont. Interaction nets. *Proceedings, 17th ACM Symposium on Principles of Programming Languages*, 95–108. ACM Press, 1990.

[3] J. Lamping. An algorithm for optimal lambda calculus reduction. *17th Annual Symposium on Principles of Programming Languages*, 16–30. ACM Press, 1990.

[4] I. Mackie, J. S. Pinto. *Compiling the Lambda Calculus into Interaction Combinators.* Logical Abstract Machines workshop, 1998.

[5] Y. Lafont. Interaction combinators. *Information and Computation*, 137(1): 69–101, 1997.

[6] D. Bechet. Universal interaction systems with only two agents. *12th International Conference on Rewriting Techniques and Applications.* Utrecht, The Netherlands, 2001.

[7] M. Fernández, I. Mackie. A calculus for interaction nets. *Proceedings of the International Conference on Principles and Practice of Declarative Programming*, vol. 1702 of *Lecture Notes in Computer Science*, 170–187. Springer-Verlag, 1999.

[8] A. Salikhmetov. *Blind graph rewriting systems.* arXiv:1204.3372, 2012.

[9] M. Fernández, I. Mackie. Operational equivalence for interaction nets. *Theoretical Computer Science*, vol. 297, 157–181.

[10] D. Mazza. Observational Equivalence for the Interaction Combinators and Internal Separation. *Electronic Notes in Theoretical Computer Science*, vol. 176, 113–137.